On Medium-Uniformity and Circuit Lower Bounds

On Medium-Uniformity and Circuit Lower Bounds

Rahul Santhanam
School of Informatics
University of Edinburgh
Edinburgh, Scotland, UK.
Email: rsanthan@inf.ed.ac.uk

Ryan Williams
Computer Science Department
Stanford University
Stanford, California, USA.
Email: rrwilliams@gmail.com

***Abstract*—We explore relationships between circuit complexity, the complexity of generating circuits, and algorithms for analyzing circuits. Our results can be divided into two parts:**

1. *Lower Bounds Against Medium-Uniform Circuits.* Informally, a circuit class is "medium uniform" if it can be generated by an algorithmic process that is somewhat complex (stronger than LOGTIME) but not infeasible. Using a new kind of indirect diagonalization argument, we prove several new unconditional lower bounds against medium uniform circuit classes, including:

- **For all k, P is not contained in P-uniform $\mathsf{SIZE}(n^k)$. That is, for all k there is a language $L_k \in \mathsf{P}$ that does not have $O(n^k)$-size circuits constructible in polynomial time. This improves Kannan's lower bound from 1982 that NP is not in P-uniform $\mathsf{SIZE}(n^k)$ for any fixed k.**
- **For all k, NP is not in $\mathsf{P}_{||}^{\mathsf{NP}}$-uniform $\mathsf{SIZE}(n^k)$. This also improves Kannan's theorem, but in a different way: the uniformity condition on the circuits is stronger than that on the language itself.**
- **For all k, $\mathsf{LOGSPACE}$ does not have $\mathsf{LOGSPACE}$-uniform branching programs of size n^k.**

2. *Eliminating Non-Uniformity and (Non-Uniform) Circuit Lower Bounds.* We complement these results by showing how to convert any potential simulation of LOGTIME-uniform NC^1 in $\mathsf{ACC}^0/\mathrm{poly}$ or $\mathsf{TC}^0/\mathrm{poly}$ into a medium-uniform simulation using small advice. This lemma can be used to simplify the proof that faster SAT algorithms imply NEXP circuit lower bounds, and leads to the following new connection:

- **Consider the following task: *given a* TC^0 *circuit* C *of* $n^{O(1)}$ *size, output* yes *when* C *is unsatisfiable, and output* no *when* C *has at least* 2^{n-2} *satisfying assignments*. (Behavior on other inputs can be arbitrary.) Clearly, this problem can be solved efficiently using randomness. If this problem can be solved deterministically in $2^{n-\omega(\log n)}$ time, then $\mathsf{NEXP} \not\subset \mathsf{TC}^0/\mathrm{poly}$.**

The lemma can also be used to derandomize randomized TC^0 simulations of NC^1 on almost all inputs:

- **Suppose $\mathsf{NC}^1 \subseteq \mathsf{BPTC}^0$. Then for every $\varepsilon > 0$ and every language L in NC^1, there is a (uniform) TC^0 circuit family of polynomial size recognizing a language L' such that L and L' differ on at most 2^{n^ε} inputs of length n, for all n.**

***Keywords*-circuit complexity, lower bounds, medium uniformity, derandomization**

I. Introduction

An important problem at the interface of "classical" computational complexity and circuit complexity is to understand the relationship between the complexity of circuits for solving problems, and the complexity of algorithmic processes that can generate those circuits from scratch. In the non-uniform setting, we put no computable bounds on the complexity of generating circuits, and in this case it is extraordinarily difficult to prove lower bounds even against "simple" circuit classes. In low-uniform[1] settings like LOGTIME-uniformity, the circuits are extremely easy to construct – circuit complexity in this regime falls in line with standard machine-based complexity classes. The "medium-uniformity" settings, where the circuit-generating process is neither too easy nor too hard, is a middle ground that is less understood, and it is quite plausible that exploring it will help us better understand the two extremes. In this paper, we study this middle ground and make concrete progress that may translate to further progress on low-uniform and non-uniform results in the future.

Lower bounds by amplifying uniformity: In the first part of the paper, we prove new lower bounds against "medium-uniform" circuits. Our key insight is that, if we assume that a medium-uniform complexity class has small medium-uniform circuits, this assumption can be applied in multiple ways: not only is it applicable to a language L in the medium-uniform class, but it is also applicable to another (medium-uniform) language encoding *circuits* for L. Applying the hypothesis multiple times allows us to simulate medium-uniformity with "low-uniformity" augmented with a small amount of advice, and diagonalize against the low-uniformity simulation to derive a contradiction.

This strategy leads to new lower bounds against notions of uniformity for which it is not possible to directly obtain lower bounds by diagonalization. Consider for example, the class of linear-size circuits. If a LOGTIME-uniformity condition is imposed on the class, then it can be simulated in nearly-linear deterministic time, and we can easily diagonalize to find a function in (for example) n^2 time that does not have such circuits. However, suppose we wish to find a function in P that does not have small circuits constructible via a "medium" uniformity notion, such as P-uniformity. Then, the notion of uniformity (which allows for an arbitrary

[1]Note that somewhat counter-intuitively, the conventional meaning of "low-uniform" is *very uniform*.

polynomial-time bound) can be more powerful than the function itself (which must lie in some fixed polynomial time bound), so it is no longer possible to directly diagonalize.

To describe our results, let us first set up some notation informally (definitions are in Section I-A). Given a class $\mathcal{C}$ of languages, recall a language L is said to have $\mathcal{C}$-uniform circuits of size $s(n)$ if there is a size-$s(n)$ circuit family $\{C_n\}$ such that the description of C_n is computable in $\mathcal{C}$. (There are several possible choices about what "description" means; in this paper, our notions of uniformity will be so powerful that these choices are all essentially equivalent.) Our first main result strengthens both the deterministic time hierarchy theorem and the classical result of Kannan [Kan82] that for any k, NP is not in P-uniform $\mathsf{SIZE}(n^k)$.

Theorem 1.1: For every k, $\mathsf{P} \not\subset \mathsf{P}$-uniform $\mathsf{SIZE}(n^k)$.

That is, for all k, there is an $L \in \mathsf{P}$ such that any algorithm generating n^k-size circuits for L must run in super-polynomial time. (Of course, the common belief is that for all k, there is an $L \in \mathsf{P}$ that does not have n^k-size circuits *at all*, but this is an extremely hard problem: resolving it would imply $\mathsf{EXP} \not\subset \mathsf{P/poly}$, for example.) The ideas in the proof of Theorem 1.1 can also be applied to smaller classes. Note that the best non-uniform branching program size lower bounds known have the form $\Omega(n^2)$ [Nec66]. However, for branching programs constructible in logarithmic space, we can prove superpolynomial lower bounds:

Theorem 1.2: For every k, $\mathsf{LOGSPACE}$ does not have $\mathsf{LOGSPACE}$-uniform branching programs of size $O(n^k)$.

We are also able to strengthen Kannan's lower bound for NP in a different direction, by *relaxing* the notion of uniformity used. To prove this result, we combine the uniformity trade-off idea used in the results above with ideas of Fortnow, Santhanam and Williams [FSW09] to show that fixed-polynomial size lower bounds can be "amplified".

Theorem 1.3: For every k, $\mathsf{NP} \not\subseteq \mathsf{P}_{||}^{\mathsf{NP}}$-uniform $\mathsf{SIZE}(n^k)$.

Reducing non-uniformity for "simple" circuit classes: The above lower bounds work by simulating medium-uniform circuits with low-uniformity and a little advice. In the second part of the paper, we study situations where *non-uniform* circuits can be simulated by "medium-uniform" ones with a little advice. We show how to translate non-uniform constant-depth circuits for NC^1 into *subexponential-time uniform* constant-depth circuits for NC^1:

Lemma 1.1: Suppose NC^1 is contained in $\mathcal{C}/\mathrm{poly}$, where $\mathcal{C} \in \{\mathsf{ACC}, \mathsf{TC}^0\}$.[2] For every $\varepsilon, k > 0$, there is a $2^{O(n^\varepsilon)}$ time and $O(n^\varepsilon)$ space algorithm that, given any circuit C of size n and depth $k \log n$, prints an $O(k/\varepsilon)$-depth, $n^{O(k)}$-size $\mathcal{C}$-circuit that is equivalent to C.

[2]In this paper, we assume by default that a complexity class $\mathcal{C}$ defined with respect to some polynomial-size class of circuits is *LOGTIME-uniform*, and we use $\mathcal{C}/\mathrm{poly}$ to denote the non-uniform version of the class. This notation makes it clear when we are discussing uniform versus non-uniform classes.

That is, a non-uniform inclusion of NC^1 in TC^0 implies small-space uniform (and hence subexponential-time uniform) TC^0 circuits for NC^1. We remark that Lemma 1.1 holds for any constant-depth circuit class over any basis (with arbitrary fan-in): ACC^0 and TC^0 just happen to be the most popular such classes.

An interesting consequence of Lemma 1.1 is that, for simulations of NC^1 in smaller classes, non-uniformity can be simulated by low-uniformity with small advice:

Corollary 1.1: For $\mathcal{C} \in \{\mathsf{ACC}, \mathsf{TC}^0\}$, $\mathsf{NC}^1 \subset \mathcal{C}/\mathrm{poly}$ if and only if for all $\varepsilon > 0$, $\mathsf{NC}^1 \subset \mathcal{C}/n^\varepsilon$.

Another consequence of Lemma 1.1 is that *randomized* simulations based on constant-depth circuits can be meaningfully derandomized, assuming they are powerful enough. For a (LOGTIME-uniform) complexity class $\mathcal{C}$, we define $\mathsf{BP}\mathcal{C}$ to be its randomized version with two-sided error in the standard way.

Theorem 1.4: Suppose $\mathsf{NC}^1 \subseteq \mathsf{BPTC}^0$. Then for every $\varepsilon > 0$ and every language L in NC^1, there is a (uniform) TC^0 circuit family of polynomial size recognizing a language L' such that L and L' differ on at most 2^{n^ε} inputs of length n, for all n.

That is, if NC^1 can be solved with uniform probabilistic TC^0 circuits, then the circuits can be made deterministic while preserving uniformity,at the expense of making errors on a small fraction of inputs. This is somewhat surprising, and it is reasonable to think that Theorem 1.4 may be applied to prove lower bounds against BPTC^0 in the future. (Currently, $\mathsf{EXP}^{\mathsf{NP}} = \mathsf{BPTC}^0$ is open!) The idea is to combine the ideas of Lemma 1.1 with prior work of Goldreich and Wigderson [GW02], who show how to derandomize algorithms in generic settings when the random bits used are both "small" with respect to the input length and "oblivious" to the input.

Lemma 1.1 can also be applied to simplify and strengthen a key component of the proof that faster $\mathcal{C}$-SAT algorithms imply $\mathsf{NEXP} \not\subset \mathcal{C}/\mathrm{poly}$ (for many circuit classes $\mathcal{C}$) [Wil10], [Wil11]. In particular, a necessary intermediate theorem says that, if there are SAT algorithms running in $O(2^n/n^{10})$ time on all polynomial-size $\mathcal{C}$-circuits, and $\mathsf{NEXP} \subset \mathcal{C}/\mathrm{poly}$, then there is a nondeterministic $o(2^n)$ time algorithm that can generate $\mathcal{C}$-circuits equivalent to a given SUCCINCT 3SAT instance. By exploiting the structure of NC^1, Lemma 1.1 can be used to deterministically generate equivalent TC^0 (respectively, ACC) circuits in *subexponential* (2^{n^ε}) time, without assuming any algorithmic improvement on circuit satisfiability.

Finally, we weaken the conditions necessary to prove lower bounds like $\mathsf{NEXP} \not\subset \mathsf{TC}^0/\mathrm{poly}$. Consider the following problem.

DERANDOMIZE-TC^0: *given a* TC^0 *circuit* C *of* $n^{O(1)}$ *size, output* yes *when* C *is unsatisfiable, and output* no *when* C *has at least* 2^{n-2} *satisfying assignments*. (Behavior on other inputs can be arbitrary.)

This problem can be trivially solved with high probability, by simply trying random assignments. We prove that a *deterministic* $2^{n-\omega(\log n)}$ time algorithm for the problem is sufficient for $\mathsf{NEXP} \not\subset \mathsf{TC}^0/\mathrm{poly}$:

Theorem 1.5: Suppose for all k, there is an $O(2^n/n^k)$ time deterministic algorithm for solving DERANDOMIZE-TC^0 on all TC^0 circuits of n inputs, n^k size, and depth k. Then $\mathsf{NEXP} \not\subset \mathsf{TC}^0/\mathrm{poly}$.

Goldreich and Meir (personal communication) have observed that the existing framework for proving NEXP circuit lower bounds from SAT algorithms also extends to derandomization problems which seem much "simpler" than SAT: for example, $2^{n-\omega(\log n)}$ time algorithms for *approximating* the acceptance probability of a given circuit to within $1/10$ is already enough to yield $\mathsf{NEXP} \not\subset \mathsf{P}/\mathrm{poly}$. The above theorem strengthens their observation even further: we may focus on the case of distinguishing unsatisfiable circuits from "very satisfiable" circuits.

A. Preliminaries and Notation

We assume basic knowledge of computational complexity [AB09]. For all complexity classes $\mathcal{C}$ defined as a class of polynomial-size circuits, we use $\mathcal{C}$ to denote the uniform version of the complexity class (with LOGTIME-uniformity being the default), and $\mathcal{C}/\mathrm{poly}$ to denote the non-uniform version. For instance, in this notation, we would say that prior work has established that $\mathsf{NEXP} \not\subset \mathsf{ACC}/\mathrm{poly}$ [Wil11]. This notation makes it clear when we are discussing uniform versus non-uniform classes.

Given a language L, L_n is the "slice" of L at length n, *i.e.*, $L_n = L \cap \{0,1\}^n$.

Given size function s and depth function d, $\mathsf{SIZE}(s)$ is the class of languages with circuits of size $O(s)$, $\mathsf{DEPTH}(d)$ the class of languages with circuits of depth d, and $\mathsf{SIZEDEPTH}(s,d)$ the class of languages which simultaneously have size $O(s)$ and depth d.

We will need standard notions of uniformity for circuits. The *direct connection language* for a sequence of circuits $C = \{C_n\}$, where C_n is on n input bits, is the language L_C consisting of all tuples of the form $\langle 1^n, g, h, r\rangle$, where g and h are indices of gates, r is the *type* of g (AND/OR/NOT/INPUT, and in case of INPUT, which of the n input bits g is, with an additional bit to specify whether g is the designated output gate), and h is a gate feeding in to g in case the type r is not INPUT. Other encodings of the direct connection language are of course possible, but for the large classes $\mathcal{C}$ we will consider, this encoding will not affect the results.

Given a class $\mathcal{C}$ of languages, a language L is said to have $\mathcal{C}$-uniform circuits of size $s(n)$ if there is a size-$s(n)$ circuit family $\{C_n\}$ such that its *direct connection language* is computable in $\mathcal{C}$. By a *description of a circuit* C_n, we mean the list of tuples in L_C corresponding to gates in C_n.

In one of our results, we also require the notion of a direct connection language for a branching program. This is defined in analogously as for a circuit, and for the notions of uniformity we use, the precise encoding will not matter.

For a uniform complexity class defined using machines or circuits, and given an advice length function $a : \mathbb{N} \to \mathbb{N}$, we incorporate advice into the class in the standard way: the machines or circuits defining the class receive an additional advice input, which depends only on the input length n, and is of length at most $a(n)$.

II. LOWER BOUNDS AGAINST MEDIUM UNIFORMITY

We will use a folklore result about a time hierarchy for deterministic time, where the lower bound holds against sublinear advice.

Proposition 1: $\mathsf{DTIME}(n^{d+1}) \not\subseteq \mathsf{DTIME}(n^d)/o(n)$, for all $d \geq 1$.

Proof: Let $\{M_i\}$ be a list of machines running in time n^d. We will construct a machine M' running in n^{d+1} time that differs from every M_i and infinite sequence of advice strings $\{a_n\}$ where $|a_n| \leq o(n)$: given an input x, $M'(x)$ simulates $M_{|x|}(x)$ augmented with advice string x', where x' is the first $|a_{|x|}|$ bits of x. ∎

The following result simultaneously strengthens the time hierarchy theorem for deterministic time [HS65], [HS66] and Kannan's result [Kan82] that for any fixed k, $\mathsf{NP} \not\subseteq \mathsf{P}\text{-uniform } \mathsf{SIZE}(n^k)$.

Reminder of Theorem 1.1: $\mathsf{P} \not\subseteq \mathsf{P}\text{-uniform } \mathsf{SIZE}(n^k)$*, for all* k*.*

Proof of Theorem 1.1: Assume $\mathsf{P} \subseteq \mathsf{P}\text{-uniform } \mathsf{SIZE}(n^k)$. Let $L \in \mathsf{P}$ be arbitrary. We will show that L can be simulated in a fixed deterministic time bound with $o(n)$ advice, which will yield a contradiction to Proposition 1.

By assumption, $L \in \mathsf{P}\text{-uniform } \mathsf{SIZE}(n^k)$, so there is a circuit family $\{C_n\}$ for L of size at most $c \cdot n^k$ for some constant c. Furthermore, by P-uniformity, the direct connection language L_{dc} for $\{C_n\}$ (see Section I-A for the definition) is in P. We consider a "succinct" version L_{succ} of the language L_{dc}, defined as follows. Letting $Bin(n)$ be the binary representation of n, define

$$L_{succ} = \{\langle Bin(n)01^{\lceil n^{1/(3k)}\rceil}, g, h, r\rangle \mid \langle 1^n, g, h, r\rangle \in L_{dc}\}.$$

Intuitively, L_{succ} is an "unpadded" version of L_{dc}.

Observe that $L_{succ} \in \mathsf{P}$. Given an input y for L_{succ}, our polynomial-time algorithm first checks if y can be parsed as a "valid" tuple $\langle z, g, h, r\rangle$, where $z = Bin(n)01^{\lceil n^{1/(3k)}\rceil}$ for some positive integer n, g and h are valid gate indices between 1 and $c \cdot n^k$, and r is a valid gate type. If this check fails, *reject*. Otherwise, the algorithm runs the polynomial-time machine deciding L_{dc} on $\langle 1^n, g, h, r\rangle$, and *accepts* if and only if this machine accepts. Note that this algorithm for L_{succ} runs in time polynomial in $|y|$, since we only simulate the machine for L_{dc} when $n^{1/(3k)} \leq |y| \leq n$ and the machine for L_{dc} runs in time polynomial in n.

Now we apply the assumption $\mathsf{P} \subseteq \mathsf{P}\text{-uniform } \mathsf{SIZE}(n^k)$ for a second time. Since $L_{succ} \in \mathsf{P}$, there is a circuit family $\{D_m\}$ of $O(m^k)$ size for L_{succ}. Given an integer n, let $m(n)$ be the least integer such the size of the tuple $\langle Bin(n)01^{\lceil n^{1/(3k)} \rceil}, g, h, r\rangle$ is at most $m(n)$ for any valid gate indices g and h for C_n and any valid gate type r. Using a standard encoding of tuples, we can assume, for large enough n, that $m(n) \leq n^{1/(2k)}$, since g, h, r can all be encoded with $O(\log n)$ bits each.

We now describe a simulation of L in time $O(n^{2k+2})$ with $o(n)$ bits of advice. Let M be an advice-taking machine which operates as follows. On input x of length n, M receives an advice string of length $O(n^{1/2} \log n)$. It interprets this advice as a circuit D_m for the language L_{succ} on inputs of length $m(n) \leq n^{1/(2k)}$. For every possible pair of gate indices g and h of C_n and every possible gate type r, M simulates the circuit D_m on $\langle Bin(n)01^{\lceil n^{1/(3k)} \rceil}, g, h, r\rangle$ to decide whether gate h is an input to gate g and whether the type of gate g is r. Each such simulation can be done in time $O(n)$, as the size of D_m is $O(n^{1/2})$. There are at most $O(n^{2k+1})$ such simulations that M performs, since there are at most that many relevant triples $\langle g, h, r\rangle$. Once all these simulations are performed, M has a full description of the circuit C_n. It then simulates C_n on x, and accepts if and only if $C_n(x)$ outputs 1. This simulation can be done in time $O(n^{2k})$ since the circuit C_n is of size $O(n^k)$. The total time taken by M is $O(n^{2k+2})$, and M uses $O(n^{1/2} \log n)$ bits of advice. By our assumptions on C_n and D_m, the simulation is correct. Thus $L \in \mathsf{DTIME}(n^{2k+2})/O(n^{1/2} \log n)$.

However, as $L \in \mathsf{P}$ was chosen to be *arbitrary*, we have $\mathsf{P} \subseteq \mathsf{DTIME}(n^{2k+2})/O(n^{1/2} \log n)$, which contradicts Proposition 1. ■

A significant property of Theorem 1.1 is that the lower bound holds for a notion of uniformity which we cannot directly diagonalize against in polynomial time. Indeed, the following proposition shows that for each d, the class we prove a lower bound against contains a language that is not in $\mathsf{DTIME}(n^d)$.

Proposition 2: For every $d \geq 1$, there is a language in $\mathsf{P}\text{-uniform } \mathsf{SIZE}(O(n))$ that is not in $\mathsf{DTIME}(n^d)$.

Proof: The standard proof of the deterministic time hierarchy theorem [HS65], [HS66] can be adapted to show that for each d, there is a unary language L which is in $\mathsf{DTIME}(n^{d+1})$ but not in $\mathsf{DTIME}(n^d)$. This unary language L can be recognized by P-uniform circuits of linear size – for each n, decide whether $1^n \in L$ in time $O(n^{d+1})$, outputting the trivial circuit which outputs the AND of its input bits if yes and the trivial circuit which outputs 0 on all inputs if no. ■

The proof ideas of Theorem 1.1 can be adapted to prove lower bounds for other classes. We next show that there for each k, there are languages in NC which do not have NC-uniform formulas of size n^k, or indeed NC-uniform circuits of fixed polynomial size and fixed polylogarithmic depth. Note that the best-known formula size lower bound in NC against non-uniform formulas is $\Omega(n^{3-o(1)})$ [Has98].

We will require a hierarchy theorem for NC, which can again be shown using standard diagonalization.

Proposition 3: For every k, NC is not contained in $\text{LOGTIME-uniform } \mathsf{SIZEDEPTH}(n^k, (\log n)^k)/o(n)$.

Theorem 2.1: For every $k \geq 1$, NC is not contained in $\mathsf{NC}\text{-uniform } \mathsf{SIZEDEPTH}(n^k, (\log n)^k))$.

Proof: Assume for a contradiction that there is a k such that $\mathsf{NC} \subseteq \mathsf{NC}\text{-uniform } \mathsf{SIZEDEPTH}(n^k, (\log n)^k)$. Let $L \in \mathsf{NC}$ be arbitrary. Let $\{C_n\}$ be a sequence of circuits of size $O(n^k)$ and depth $(\log n)^k$ solving L, and $L_{dc} \in \mathsf{NC}$ be the direct connection language of $\{C_n\}$. Define the succinct version L_{succ} of L_{dc} as in the proof of Theorem 1.1, with the same parameter $m(n)$. Observe that $L_{succ} \in \mathsf{NC}$ since checking whether a "succinct" tuple is valid, and then converting to a full tuple that can be offered as input to L_{dc} are both procedures that can be implemented in polylogarithmic depth. Hence L_{succ} has circuits of depth $O(m^k)$ and depth $O((\log m)^k)$, by assumption.

Now we will define a family of LOGTIME-uniform $\mathsf{SIZEDEPTH}(n^{k'}, (\log n)^{k'})$ circuits taking $o(n)$ bits of advice which decide if $x \in L$, where k' is a fixed constant depending on k. The circuits interpret the advice as small-depth circuits for L_{succ} on inputs of length $m(n)$. The circuits simulate C_n on x implicitly, running the small-depth circuit for L_{succ} to retrieve any bit of C_n that is required. Since $m(n) \leq n^{1/(2k)}$, each run of the small-depth circuit for L_{succ} incurs a depth cost at most $O((\log n)^k)$ and size cost at most $n^{2/3}$. Simulating a circuit of depth $O((\log n)^k)$ and size $O(n^k)$ on an input can be done uniformly in size $O(n^{2k})$ and depth $O((\log n)^k)$. Because the circuit is being simulated implicitly, we incur an additional cost in size and depth, but the overall size is at most $O(n^{3k})$ and depth at most $O((\log n)^{k^2})$. Thus, by setting $k' = k^2$, we have the required simulation. But this contradicts Proposition 3, since L is arbitrary. ■

Similarly, the following can be shown. We omit the proof because of its similarity to the previous ones.

Reminder of Theorem 1.2: For any k, LOGSPACE *does not have* LOGSPACE*-uniform branching programs of size* $O(n^k)$.

Theorem 1.1 improves Kannan's old result that $\mathsf{NP} \not\subseteq \mathsf{P}\text{-uniform } \mathsf{SIZE}(n^k)$ by showing a better upper bound for the hard language, i.e., P rather than NP. We can improve his result in a different way by relaxing the uniformity condition instead to $\mathsf{P}_{||}^{\mathsf{NP}}$-uniformity. The main idea is to first relativize Theorem 1.1 to allow parallel access to an NP oracle both in the upper bound and in the uniformity bound, and then to strengthen the upper bound using an idea of Fortnow, Santhanam and Williams [FSW09].

Reminder of Theorem 1.3: Let $k \geq 1$ be any constant. Then $\mathsf{NP} \not\subseteq \mathsf{P}_{||}^{\mathsf{NP}}\text{-uniform } \mathsf{SIZE}(n^k)$.

Proof: First we claim that $\mathsf{P}_{||}^{\mathsf{NP}}$ is not contained in $\mathsf{P}_{||}^{\mathsf{NP}}$-uniform $\mathsf{SIZE}(n^k)$; the proof is completely analogous to that of Theorem 1.1. Then we claim that $\mathsf{P}_{||}^{\mathsf{NP}} \not\subseteq \mathsf{P}_{||}^{\mathsf{NP}}$-uniform $\mathsf{SIZE}(n^k)$ implies that $\mathsf{NP} \not\subseteq \mathsf{P}_{||}^{\mathsf{NP}}$-uniform $\mathsf{SIZE}(n^{k-1})$. This result was shown without the uniformity conditions by Fortnow, Santhanam and Williams [FSW09]. An examination of their proof shows that a circuit C_n for any language in $\mathsf{P}_{||}^{\mathsf{NP}}$ can be constructed using fixed polynomial-time oracle access to circuits for two specific languages in NP. If each of the circuit sequences for these languages is $\mathsf{P}_{||}^{\mathsf{NP}}$-uniform then so is the small circuit sequence for the $\mathsf{P}_{||}^{\mathsf{NP}}$-uniform language, by converting the polynomial-time oracle machine to an oracle circuit and then substituting the circuits for the two oracles. Since k is arbitrary, we are done. ∎

For $\mathsf{E} = \mathsf{DTIME}(2^{O(n)})$, we do not get an unconditional lower bound, but rather a "gap result" in the style of Impagliazzo and Wigderson [IW01] or Buresh-Oppenheim and Santhanam [BOS06]. The result states that if we can diagonalize in E against an arbitrarily small exponential amount of advice, then we get lower bounds against E-uniform circuits of size close to the best possible. The main idea is to use the proof idea of Theorem 1.1 recursively.

Theorem 2.2: If $\mathsf{E} \not\subseteq \mathsf{DTIME}(2^{2n})/2^{\epsilon n}$ for some $\epsilon > 0$, then $\mathsf{E} \not\subseteq \mathsf{E}$-uniform $\mathsf{SIZE}(2^{\delta n})$ for any $\delta < 1$.

Proof: Let $\delta < 1$ be any constant. Assume that for any $L \in \mathsf{E}$, $L \in \mathsf{E}$-uniform $\mathsf{SIZE}(2^{\delta n})$. We will show that it follows that $L \in \mathsf{DTIME}(2^{2n})/2^{\epsilon n}$ for any constant $\epsilon > 0$.

We define a sequence of languages L_i as follows. $L_0 = L$. In general, L_i will be a "succinct" version of a connection language of circuits for L_{i-1}. The direct connection language we used before will not be succinct enough for our purposes, so we use instead what we call the indirect connection language L_{ic} of a sequence of circuits, where a tuple $\langle 1^n, g, i, b_1, b_2, r\rangle$ is in L_{ic} iff the gate with index g has type r, and moreover, if the type r is not INPUT, then if the bit $b_1 = 0$, the i'th bit of the index of the first input to g is b_2, and if the bit $b_1 = 1$, the i'th bit of the index of the second input to g is b_2. Essentially, L_{ic} encodes the adjacency list corresponding to the DAGs of the circuit sequence rather than the adjacency matrix. Note that for any sequence of circuits of size $2^{O(n)}$, $L_{ic} \in E$ iff $L_{dc} \in E$.

We now define L_1 more precisely. By assumption, there is a sequence of circuits $\{C_n\}$ for L such that C_n is of size $O(2^{\delta n})$ for each n, and the indirect connection language $L_{ic,0}$ of the sequence of circuits can be decided in E (since by assumption, the direct connection language can be decided in E). L_1 is the succinct version of $L_{ic,0}$ defined as follows: a tuple $\langle Bin(n), g, i, b_1, b_2, r\rangle$ belongs to L_1 iff the tuple $\langle 1^n, g, i, b_1, b_2, r\rangle$ belongs to $L_{ic,0}$. Note that since the gate index of g requires at least δn bits to describe, we can decide L_1 in E, hence by assumption, L_1 has E-uniform circuits of size $2^{\delta m}$.

Let $\{C_m^1\}$ be an E-uniform sequence of circuits of size $2^{\delta m}$ for L_1. As a function of n, the size of C_m^1 is at most $O(2^{\delta(\delta n + O(\log n))}) = O(2^{\delta^2 n}\mathrm{poly}(n))$. Let $L_{ic,1}$ be the indirect connection language of the sequence $\{C_m^1\}$. We define L_2 to be the succinct version of $L_{ic,1}$ completely analogously to the previous paragraph.

Continuing in this way, we get a sequence of languages $L_1, L_2 \ldots$ such that L_k has E-uniform circuits of size $O(2^{\delta^k n}\mathrm{poly}(n))$. Let k be such that $\delta^k < \epsilon$. Since $\delta < 1$, there exists such a k.

We now define a simulation of L in time $O(2^{2n})$ with $O(2^{\epsilon n})$ bits of advice. The advice is the description of a circuit for L_k. Given this description, we can recover in time $2^{(\delta^{k-1}+\delta^k+o(1))n}$ the description of a circuit for L_{k-1}. Again, from this description, we can recover in time $2^{(\delta^{k-1}+\delta^{k-2}+o(1))n}$ the description of a circuit for L_{k-2}. Continuing in this way, we can recover in total time $2^{(\delta+\delta^2+o(1))n} = O(2^{2n})$ the circuit C_n for L, whereupon we can run C_n on L to determine whether the input belongs to L or not.

∎

Note that the conditional lower bound of Theorem 2.2 is close to best possible, as shown by the following easy result.

Proposition 4: E has E-uniform circuits of size at most $n2^n$.

Proof: For any language L in E, the truth table of L can be computed in linear exponential time, and from the truth table it is easy to compute canonical DNFs or CNFs of size at most $n2^n$ for L. ∎

III. A Uniformization Lemma For NC^1

We now turn to the problem of eliminating non-uniformity in low-complexity circuit classes. Recall the Formula Eval problem: *given a formula F and input v to it, determine whether $F(v) = 1$.* Buss [Bus87] showed that Formula Eval is complete under LOGTIME-reductions for LOGTIME-uniform NC^1. Hence Formula Eval can be solved efficiently (in, for example, TC^0) iff $\mathsf{NC}^1 \subseteq \mathsf{TC}^0$.

Theorem 3.1: Suppose $\mathsf{NC}^1 \subset \mathcal{C}/\mathrm{poly}$, where $\mathcal{C} \in \{\mathsf{ACC}, \mathsf{TC}^0\}$. For every $\varepsilon > 0$, there is a $2^{O(n^\varepsilon)}$ time and $O(n^\varepsilon)$ space algorithm that, given 1^n, prints an $O(1/\varepsilon)$-depth, $n^{O(1)}$-size $\mathcal{C}$-circuit that solves Formula Eval on formulas of size n.

The following lemma is an immediate corollary:

Reminder of Lemma 1.1: Suppose $\mathsf{NC}^1 \subset \mathcal{C}/\mathrm{poly}$, where $\mathcal{C} \in \{\mathsf{ACC}, \mathsf{TC}^0\}$. For every $\varepsilon, k > 0$, there is a $2^{O(n^\varepsilon)}$ time and $O(n^\varepsilon)$ space algorithm that, given any circuit C of size n and depth $k \log n$, prints an $O(k/\varepsilon)$-depth, $n^{O(k)}$-size $\mathcal{C}$-circuit that is equivalent to C.

The proof is inspired by Allender and Koucký [AK10] who showed that if $\mathsf{NC}^1 \subset \mathcal{C}/\mathrm{poly}$, then the problem Balanced Formula Evaluation has $n^{1+\varepsilon}$ size $\mathcal{C}$-circuits, for $\mathcal{C} \in \{\mathsf{ACC}, \mathsf{TC}^0\}$. Rather than focusing on reducing circuit sizes, we focus on reducing non-uniformity.

Proof of Theorem 3.1: Assuming $\mathsf{NC}^1 \subset \mathcal{C}/\text{poly}$, let $k \geq 1$ be such that the FORMULA EVAL problem on formulas of size n has $\mathcal{C}$-circuits of n^k size. Buss [Bus87] showed that FORMULA EVAL can be solved in LOGTIME-uniform NC^1. Applying this algorithm, we can generate (in polynomial time) an n^c size formula $G(F,x)$ such that $G(F,x) = 1 \iff F(x) = 1$, for all formulas F of size n and all potential inputs x of length up to n. WLOG, G has depth at most $c \log n$, for some fixed $c \geq 1$.

Partition G into $t = n^{c-\varepsilon/k}$ subformulas $F_1, \ldots, F_t$ of at most $n^{\varepsilon/k}$ gates each. More precisely, we break the $c \log n$ levels of G into kc/ε groups, where each group contains $(\varepsilon/k)\log n$ adjacent levels. Each group consists of subformulas of depth $(\varepsilon/k)\log n$ and size at most $n^{\varepsilon/k}$. WLOG, we may assume each F_i has the same number of inputs (roughly $n^{\varepsilon/k}$).

Next, we "brute force" a small $\mathcal{C}$ circuit for small instances of FORMULA EVAL. Try all possible $\mathcal{C}$ circuits of size n^ε for FORMULA EVAL on all formula-input pairs of length up to $n^{\varepsilon/k}$. For each trial circuit T, we try all possible $2^{\tilde{O}(n^{\varepsilon/k})}$ formula-input pairs, and check that T correctly evaluates the input on the formula. By our choice of k, at least one T will pass this check on all of its inputs.

Once a suitable T has been found, we replace every subformula $F_i(y_1, \ldots, y_q)$ in G with the $\mathcal{C}$ circuit $T(F_i, y_1, \ldots, y_q)$. By our choice of T, the resulting circuit is equivalent to G, has size $O(n^{c-\varepsilon/k} \cdot n^\varepsilon) \leq O(n^{c+\varepsilon})$, and has depth dkc/ε, where d is the depth of T.

It is clear that the algorithm can run in $2^{O(n^\varepsilon)}$ time for any $\varepsilon > 0$. Furthermore, it can also be implemented to run in $O(n^\varepsilon)$ space: any desired bit of the n^c size formula G for FORMULA EVAL instance can be generated in LOGTIME, so the brute-force search for an n^ε size $\mathcal{C}$ circuit T equivalent to FORMULA EVAL can be carried out in $O(n^\varepsilon)$ space. Given T, the rest of the $\mathcal{C}$ can easily be generated in $O(n^\varepsilon)$ space by reading the appropriate bits from G. ∎

Note that, rather than brute-forcing the small $\mathcal{C}$ circuit for FORMULA EVALUATION, we could have simply provided it as advice. This implies:

Reminder of Corollary 1.1: For $\mathcal{C} \in \{\mathsf{ACC}, \mathsf{TC}^0\}$, $\mathsf{NC}^1 \subset \mathcal{C}/\text{poly} \iff$ *for all* $\varepsilon > 0$, $\mathsf{NC}^1 \subset \mathcal{C}/n^\varepsilon$.

Lemma 1.1 and Corollary 1.1 have consequences for lower bounds as well as algorithms. We first give a consequence for lower bounds, showing that either TC^0 computations cannot be speeded up in general using logarithmic depth and bounded fan-in, or NC^1 does not have non-uniform polynomial-size threshold circuits of bounded depth.

We need a hierarchy theorem for TC^0, which can be shown analogously to Proposition 1 and Proposition 3.

Proposition 5: For every constant k and d and all $\epsilon < 1$, there is a language in TC^0 which cannot be decided by LOGTIME-uniform TC^0 circuits of size n^k and depth d with n^ϵ bits of advice.

Theorem 3.2: At least one of the following holds:

- For all constants k, there is a language in TC^0 which does not have LOGTIME-uniform circuits of depth $k \log n$.
- $\mathsf{NC}^1 \not\subset \mathsf{TC}^0/\text{poly}$.

Proof: Assume that $\mathsf{NC}^1 \subset \mathsf{TC}^0/\text{poly}$ and that there is a constant k such that each language in TC^0 has LOGTIME-uniform circuits of depth $k\log(n)$. We derive a contradiction.

Let $L \in \mathsf{TC}^0$. By the second assumption, L has LOGTIME-uniform circuits of depth $k\log(n)$. From the first assumption and using Corollary 1.1 with $\epsilon = 1/(2k)$, we have that there exists constants c and d such that FORMULA EVAL can be decided by uniform threshold circuits of size m^c and depth d with m^ϵ bits of advice. This implies that any language with LOGTIME-uniform circuits of depth $k\log(n)$ can be decided by uniform threshold circuits of size $O(n^{kc})$ and depth d with $O(n^{1/2})$ bits of advice, and hence so can L. Since L is an arbitrary language in TC^0, this contradicts Proposition 5. ∎

We can also use Lemma 1.1 to derive algorithmic consequences of $\mathsf{NC}^1 \subset \mathsf{ACC}/\text{poly}$. Practically anything computable in subexponential time on ACC circuits can be extended to NC^1 circuits, under the assumption. For instance:

Corollary 3.1: If $\mathsf{NC}^1 \subset \mathsf{ACC}/\text{poly}$ then for all c, satisfiability of n^c size formulas with n variables can be computed (deterministically) in $O(2^{n-n^\varepsilon})$ time, for some $\varepsilon > 0$ depending on c.

Proof: Given a formula F of size n^c, apply Lemma 1.1 to generate an equivalent $n^{O(c/\delta)}$ size ACC circuit of depth $O(1/\delta)$, in $2^{O(n^\delta)}$ time, for some $\delta < 1$. Satisfiability of the ACC circuit can be determined in 2^{n-n^ε} time via an ACC-SAT algorithm [Wil11]. ∎

A. Derandomizing TC^0 *by Assuming Randomized* TC^0 *is Powerful*

Next, we prove that if NC^1 can be simulated in randomized TC^0, then we can derive a non-trivial *deterministic* TC^0 simulation of NC^1 as well.

Reminder of Theorem 1.4: Suppose $\mathsf{NC}^1 \subseteq \mathsf{BPTC}^0$. *Then for every* $\varepsilon > 0$ *and every language* L *in* NC^1, *there is a (uniform)* TC^0 *circuit family of polynomial size recognizing a language* L' *such that* L *and* L' *differ on at most* 2^{n^ε} *inputs of length* n, *for all* n.

Proof: Assume $\mathsf{NC}^1 \subseteq \mathsf{BPTC}^0$. It follows that $\mathsf{NC}^1 \subset \mathsf{TC}^0/\text{poly}$, and therefore by the arguments of Lemma 1.1, by providing n^ε advice (namely, a small TC^0 circuit for evaluating arbitrary formulas of size $n^{\varepsilon/k}$) we can translate any n^c size NC^1 circuit into a $n^{O(c)}$ size TC^0 circuit of depth $O(1/\varepsilon)$, in polynomial time.

However, the assumption that $\mathsf{NC}^1 \subseteq \mathsf{BPTC}^0$ yields more: rather than n^ε bits of *non-uniform* advice, the inclusion provides a LOGTIME-uniform TC^0 circuit $C(F,r)$ of size n^ε, which takes a formula F of size $n^{\varepsilon/k}$, an input x

of size $n^{\varepsilon/k}$ and at most n^{ε} bits of randomness r as input, such that for every F and x,

$$\Pr_{r\in\{0,1\}^{n^{\varepsilon}}}[C(F,x,r)=F(x)]>3/4.$$

We first need to amplify the success probability of the circuit C. Let t be a parameter, and define a new TC^0 circuit $C'(F,r_1,\ldots,r_t)$ which takes the MAJORITY of $C(F,r_i)$ for $i=1,\ldots t$. By standard probabilistic arguments, for every F and x we have

$$\Pr_{r_1,\ldots,r_t\in\{0,1\}^{n^{\varepsilon}}}[C'(F,x,r_1,\ldots,r_t)=F(x)]>1-2^{\Omega(t)}.$$

Choose $t=d\cdot n^{\varepsilon}$ for sufficiently large d, so that the probability of agreement is greater than $1-1/2^{3n^{\varepsilon}}$. Then by a union bound, we have that random choices of $r_1,\ldots,r_t$ are simultaneously good for *all* F and x of at most $n^{\varepsilon/k}$ size, i.e.,

$$\Pr_{r_i\in\{0,1\}^{n^{\varepsilon}}}[(\forall\ F,x)\ C'(F,x,r_1,\ldots,r_t)=F(x)]>1-2^{n^{\varepsilon}}.$$

Now for a given NC^1 circuit N of size n^c, after converting N into a formula F_N of size $S=n^{O(c)}$ there will be at most $2S/n^{\varepsilon/k}$ subformulas of F_N, each of $n^{\varepsilon/k}$ size, which need to be accurately modeled by the TC^0 circuit C'. Replace each $n^{\varepsilon/k}$-size subformula $F'(x')$ of F_N with $C'(F',x',r_1,\ldots,r_t)$, and let D be the TC^0 circuit that results from this replacement. Since C' succeeds against *all* formulas F and inputs x of size at most $n^{\varepsilon/k}$ with high probability, we conclude that in fact our circuit D works for all inputs (of length n) with high probability, i.e.,

$$\Pr_{r_1,\ldots,r_t\in\{0,1\}^{n^{\varepsilon}}}[(\forall\ x \text{ of length } n)\ D(x,r_1,\ldots,r_t)=N(x)]>1-2^{n^{\varepsilon}}.$$

Therefore, using only $d\cdot n^{2\varepsilon}$ random bits $r_1,\ldots,r_t$, we can efficiently construct a TC^0 circuit D that agrees with N on a given input. That is, we are in a situation where short, sublinear-length and randomly chosen "advice" (chosen *prior* to receiving the input x) succeeds against all inputs x simultaneously, with high probability.

This is precisely the situation described in a paper of Goldreich and Wigderson [GW02] who prove that in such situations, one can generically provide, for every $\varepsilon>0$, a *deterministic* simulation which is successful on all but $2^{n^{\varepsilon}}$ inputs of length n, by extracting additional randomness from the input itself. In more detail, say that a function $E_n:\{0,1\}^n\times\{0,1\}^m\to\{0,1\}^{\ell}$ is a *k-extractor* if, for every random variable X over $\{0,1\}^n$ that puts probability mass at most $1/2^k$ on all inputs,[3] the random variable $E(X,U_m)$ has statistical distance at most $1/10$ from U_{ℓ} (where U_q denotes the uniform distribution on $\{0,1\}^q$). Given a randomized algorithm R using ℓ bits of randomness, and an extractor E_n which takes n bits and $e\log n$ bits and outputs ℓ bits, Goldreich and Wigderson's deterministic simulation of R is simply:

> Given an input x of length n, output the majority value of $R(x,E_n(x,r))$ over all binary strings r of length $e\log n$.

Goldreich and Wigderson (Theorem 3, [GW02]) prove that, when we use a family of functions

$$\{E_n:\{0,1\}^n\times\{0,1\}^{e\log n}\to\{0,1\}^{\ell(n)}\}_n$$

such that E_n is a $k(n)$-extractor for all n, the above algorithm agrees with R on all but $2^{k(n)}$ of the n-bit inputs. To complete the proof, it suffices for us to exhibit an extractor family $\{E_n\}$ that is computable in LOGTIME-uniform TC^0 and has $k(n)=\ell(n)^c$ for a fixed constant c. Then, our final uniform TC^0 simulation of C will compute the MAJORITY value of $D(x,E_{|x|}(x,r))$ over all $O(n^e)$ random seeds r. In our case, the amount of randomness needed can be made $\ell(n)=n^{\varepsilon}$ for any desired $\varepsilon>0$, so this simulation will err on at most $2^{k(n)}\leq 2^{n^{c\varepsilon}}$ inputs for a fixed c and arbitrarily small $\varepsilon>0$.

Finally, we observe that such extractors do exist: indeed, the extractors corresponding to Impagliazzo-Wigderson pseudorandom generators provided by Theorem 5 in Trevisan [Tre01] are computable in LOGTIME-uniform TC^0, by using Theorem 4.6 in Viola [Vio05]. ■

B. Very Weak Derandomization For TC^0 Lower Bounds

Lemma 1.1 also has bearing on the emerging connections between circuit satisfiability algorithms and circuit lower bounds. We can give a much simpler proof that faster TC^0 SAT algorithms imply $\mathsf{NEXP}\not\subset\mathsf{TC}^0/\text{poly}$ (originally proved in [Wil11] with a more involved argument):

Theorem 3.3: Suppose for all k, there is an $O(2^n/n^{10})$ time algorithm for solving satisfiability of TC^0 circuits with n inputs, n^k size, and depth k. Then $\mathsf{NEXP}\not\subset\mathsf{TC}^0/\text{poly}$.

Proof: (Sketch) In their work on succinct PCPs, Ben-Sasson *et al.* [BGHSV05] show that for any $L\in\mathsf{NTIME}[2^n]$ and instance x of length n, one can generate (in $\text{poly}(n)$ time) an NC^1 circuit C_x with $n+c\log n$ inputs and n^c size for a universal $c<10$, such that $x\in L$ if and only if the truth table of C_x encodes a satisfiable constraint satisfaction problem, where each constraint has $O(1)$ variables.

Williams [Wil10] (Theorem 3.4) shows how to use this result to prove that if NC^1 circuit satisfiability on all n-input n^k-size circuits is solvable in $O(2^n/n^{10})$ time, then $\mathsf{NEXP}\not\subset\mathsf{NC}^1/\text{poly}$. Briefly, the idea is to assume that the circuit SAT algorithm exists and that $\mathsf{NEXP}\subset\mathsf{NC}^1/\text{poly}$, and use the two assumptions to nondeterministically simulate an arbitrary $L\in\mathsf{NTIME}[2^n]$ in nondeterministic $o(2^n)$ time (a contradiction to the nondeterministic time

[3] More formally, for all $x\in\{0,1\}^n$, $\Pr[X=x]\leq 1/2^k$.

hierarchy).[4] This simulation of an arbitrary L can be done by constructing the aforementioned NC^1 circuit C_x on the input x, guessing an NC^1 circuit C' that encodes a satisfying assignment (i.e., a witness) to the constraint satisfaction problem, then composing C_x and C' to form an NC^1 circuit D which is unsatisfiable if and only if the truth table of C' is a satisfying assignment. An NC^1 circuit SAT algorithm that takes $O(2^{n+c\log n}/(n+c\log n)^{10}) = o(2^n)$ time results in a contradiction.

Assume now that $\mathsf{NEXP} \subset \mathsf{TC}^0/\mathrm{poly}$ and we have an algorithm A for TC^0 circuit SAT meeting the hypothesis of the theorem. We wish to derive a contradiction. The first assumption, along with Lemma 1.1, implies there is a deterministic subexponential time algorithm B which, given an NC^1 circuit D, can generate an equivalent TC^0 circuit E only polynomially larger than D. Therefore, we can solve NC^1 circuit SAT in $O(2^n/n^{10})$ time as well, by applying the algorithm B to convert a given NC^1 circuit into TC^0, then applying algorithm A for TC^0 circuit SAT. By the previous paragraph, this implies that $\mathsf{NEXP} \not\subset \mathsf{NC}^1/\mathrm{poly}$, a contradiction (as TC^0 is contained in NC^1). ■

We can also show that very weak derandomization of TC^0 suffices for such lower bounds. For $\mathcal{C} \in \{\mathsf{P}, \mathsf{NC}^1, \mathsf{TC}^0\}$, define DERANDOMIZE-$\mathcal{C}$ to be the problem: *given a* $\mathcal{C}/\mathrm{poly}$ *circuit* C*, output* yes *when* C *is unsatisfiable and* no *when* C *has at least* 2^{n-2} *satisfying assignments, with arbitrary behavior otherwise*. We say that a nondeterministic algorithm A *solves* DERANDOMIZE-TC^0 if for all circuits C,

- every computation path of $A(C)$ leads to one of three states: *reject*, *unsatisfiable*, or *satisfying*,
- at least one path of $A(C)$ does not lead to *reject*,
- if C has at least 2^{n-2} satisfying assignments then no path of $A(C)$ is *unsatisfiable*,
- if C is unsatisfiable then no path of $A(C)$ is *satisfiable*.

Reminder of Theorem 1.5: Suppose for all k*, there is an* $O(2^n/n^k)$ *time algorithm for solving* DERANDOMIZE-TC^0 *on all* TC^0 *circuits of* n *inputs,* n^k *size, and depth* k*. Then* $\mathsf{NEXP} \not\subset \mathsf{TC}^0/\mathrm{poly}$.

Proof: (Sketch) Our proof will be by contradiction, as in Theorem 3.3. The succinct PCPs of Ben-Sasson *et al.* [BGHSV05] can be used to show that if DERANDOMIZE-P on all n-input n^k-size circuits can be solved in $O(2^n/n^{10})$ time, then $\mathsf{NEXP} \not\subset \mathsf{P}/\mathrm{poly}$ ([Wil10]). An improvement of [BGHSV05] given by Thilo Mie [Mie09] yields succinct PCPs with $O(1)$ queries. That is, given any $L \in \mathsf{NTIME}[2^n]$ and instance x of length n, one can generate (in $\mathrm{poly}(n)$ time) a circuit C_x with $n + c\log n$ inputs and n^c size,[5] such that

- if $x \in L$ then the truth table of C_x encodes a satisfiable k-CSP (for some constant k), and
- if $x \notin L$ then the truth table of C_x encodes a k-CSP such that every variable assignment satisfies at most 10% of the constraints.

Say that a circuit is *at most* ρ*-satisfiable* (respectively, at least ρ-satisfiable) if it has at most (at least) $\rho \cdot 2^n$ satisfying assignments. Analogously to the proof of Theorem 3.3, the above can be used to (nondeterministically) produce a polynomial-size circuit C' on $n + c\log n$ inputs which is always either unsatisfiable or at least $9/10$-satisfiable, such that determining which is the case in $o(2^n)$ nondeterministic time yields a contradiction to $\mathsf{NEXP} \not\subset \mathsf{P}/\mathrm{poly}$. (More precisely, given an instance x of L, we construct a new nondeterministic algorithm for L, which guesses a circuit D intended to encode a satisfying assignment A to the k-CSP encoded by C_x, and verifies this guess by constructing a circuit C' of polynomially larger size which contains copies of C_x and k copies of D, such that $C'(i) = 0$ if and only if the ith constraint of the k-CSP is satisfied by the assignment A. This construction appears in [Wil10] so we will not repeat it here. By properties of the PCP, C' is either unsatisfiable or at least $9/10$-satisfiable. With an $O(2^n/n^{10})$ time algorithm that can distinguish the two cases, N can recognize L in $o(2^n)$ time.) We will show that a slightly faster algorithm for *Derandomize*-TC^0 is enough to yield a contradiction to $\mathsf{NEXP} \not\subset \mathsf{TC}^0/\mathrm{poly}$.

Assuming $\mathsf{P} \subset \mathsf{TC}^0/\mathrm{poly}$, the generated circuits C' have equivalent TC^0 circuits D of polynomially larger size. We will construct a nondeterministic $o(2^n)$ time algorithm N recognizing an arbitrary $L \in \mathsf{NTIME}[2^n]$. The algorithm N starts by guessing a TC^0 circuit D' that given (x, i) prints the output of the ith gate of $C'(x)$, where i ranges from 1 to the size of C'. (Such a D' also exists, under the assumption.) Then N sets up a TC^0 circuit E (of $\mathrm{poly}(n)$ size) such that $E(x) = 0$ if and only if for all gates i of C', $D(x, i)$ is *consistent*: that is, if i_1 and i_2 are indices to the gates that are inputs to gate i, then the output value $D(x, i)$ is consistent with the input values $D'(x, i_1)$ and $D'(x, i_2)$. (The construction of E is analogous to Lemma 3.1 in [Wil11].) Letting $i^\star$ be the index of the output gate of C', we have that $E(x) = 0$ implies $D'(x, i^\star) = C'(x)$.

Let A be an algorithm that outputs *no* if a given TC^0 circuit is at least $1/4$-satisfiable, and *yes* if the circuit is unsatisfiable. N runs A on E, and *rejects* if A returns *no*. Next, N runs A on the circuit $E'(x) := D'(x, i^\star)$, and

[4] Technically speaking, Theorem 3.4 in [Wil10] only shows the consequence $\mathsf{E}^{\mathsf{NP}} \not\subset \mathsf{NC}^1/\mathrm{poly}$, but this can be easily improved to $\mathsf{NEXP} \not\subset \mathsf{NC}^1/\mathrm{poly}$, by extending work of Impagliazzo, Kabanets, and Wigderson [IKW02] to show that $\mathsf{NEXP} \subset \mathsf{NC}^1/\mathrm{poly}$ implies every problem in NEXP has witnesses that can be encoded as truth tables of NC^1 circuits.

[5] We believe that the circuit C_x in Mie's construction can be made to have $O(\log n)$ depth as well, which would simplify our argument: we could assume that C_x is NC^1, then apply the strategy of Theorem 3.3. However, verifying this is quite technically involved, so we leave it as an interesting open problem here, and provide an alternative argument.

accepts if and only if A returns *yes*. This concludes the description of N.

To verify that this algorithm N is correct, first suppose C' is unsatisfiable. If N guesses a TC^0 circuit D' consistent with C', then E is unsatisfiable and A returns *yes* on E. The circuit E' must be unsatisfiable as well, so A returns *yes* on E' and N accepts on some path.

For the other case, suppose C is at least $9/10$-satisfiable, and N guesses some circuit D' and constructs E. If A returns *no* on E then the computation path rejects. Otherwise, if A returns *yes* on E, then by assumption, E is at most $1/4$-satisfiable, so $E(x) = 1$ on at most $1/4$ of all possible x. Therefore, $C(x) = D'(x, i^\star)$ on at least $3/4$ of all x. In the worst case, $D'(x, i^\star) = 1$ on at least $(9/10 - 1/4)$ of the possible inputs, hence E' is at least $(9/10 - 1/4)$-satisfiable, algorithm A returns *no* on E', and N rejects on these computation paths as well. ■

Acknowledgements: We thank Eli Ben-Sasson for useful discussions on the nice properties of known succinct PCPs. R.W. was supported in part by a David Morgenthaler II Faculty Fellowship at Stanford, and NSF CCF-1212372. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

REFERENCES

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009.

[AK10] Eric Allender and Michal Koucký. Amplifying lower bounds by means of self-reducibility. *J. ACM*, 57(3):14:1–14:36, 2010.

[BGHSV05] Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil P. Vadhan. Short PCPs verifiable in polylogarithmic time. In *IEEE Conference on Computational Complexity*, pages 120–134, 2005.

[BOS06] Joshua Buresh-Oppenheim and Rahul Santhanam. Making hard problems harder. In *IEEE Conference on Computational Complexity*, pages 73–87, 2006.

[Bus87] Samuel R. Buss. The Boolean formula value problem is in ALOGTIME. In *STOC*, pages 123–131, 1987.

[FSW09] Lance Fortnow, Rahul Santhanam, and Ryan Williams. Fixed polynomial size circuit bounds. In *IEEE Conference on Computational Complexity*, pages 19–26, 2009.

[GW02] Oded Goldreich and Avi Wigderson. Derandomization that is rarely wrong from short advice that is typically good. In *Proceedings of the 6th International Workshop on Randomization and Approximation Techniques in Computer Science*, pages 209–223, 2002.

[Has98] Johan Hastad. The shrinkage exponent of de Morgan formulas is 2. *SIAM Journal on Computing*, 27(1):48–64, 1998.

[HS65] Juris Hartmanis and Richard Stearns. On the computational complexity of algorithms. *Trans. Amer. Math. Soc. (AMS)*, 117:285–306, 1965.

[HS66] Frederick Hennie and Richard Stearns. Two-tape simulation of multitape Turing machines. *Journal of the ACM*, 13(4):533–546, October 1966.

[IKW02] Russell Impagliazzo, Valentine Kabanets, and Avi Wigderson. In search of an easy witness: Exponential time vs. probabilistic polynomial time. *Journal of Computer and System Sciences*, 65(4):672–694, 2002.

[IW01] Russell Impagliazzo and Avi Wigderson. Randomness vs time: derandomization under a uniform assumption. *Journal of Computer and System Sciences*, 63(4):672–688, 2001.

[Kan82] Ravi Kannan. Circuit-size lower bounds and non-reducibility to sparse sets. *Information and Control*, 55(1):40–56, 1982.

[Mie09] Thilo Mie. Short PCPPs verifiable in polylogarithmic time with $O(1)$ queries. *Ann. Math. Artif. Intell.*, 56(3-4):313–338, 2009.

[Nec66] Eduard Neciporuk. On a boolean function. *Doklady of the Academy of the USSR*, 169(4):765–766, 1966.

[Tre01] Luca Trevisan. Extractors and pseudorandom generators. *Journal of the ACM*, 48(4):860–879, 2001.

[Vio05] Emanuele Viola. The complexity of constructing pseudorandom generators from hard functions. *Computational Complexity*, 13(3):147–188, 2005.

[Wil10] Ryan Williams. Improving exhaustive search implies superpolynomial lower bounds. In *STOC*, pages 231–240, 2010.

[Wil11] Ryan Williams. Non-uniform ACC circuit lower bounds. In *IEEE Conference on Computational Complexity*, pages 115–125, 2011.